Network Working Group
Request for Comments: 3029
Category: Experimental

C. Adams
Entrust Technologies
P. Sylvester
EdelWeb SA - Groupe ON-X Consulting
M. Zolotarev
Baltimore Technologies Pty Limited
R. Zuccherato
Entrust Technologies

February 2001

Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes a general Data Validation and Certification Server (DVCS) and the protocols to be used when communicating with it. The Data Validation and Certification Server is a Trusted Third Party (TTP) that can be used as one component in building reliable non-repudiation services.

Useful Data Validation and Certification Server responsibilities in a PKI are to assert the validity of signed documents, public key certificates, and the possession or existence of data.

Assertions created by this protocol are called Data Validation Certificates (DVC).

We give examples of how to use the Data Validation and Certification Server to extend the lifetime of a signature beyond key expiry or revocation and to query the Data Validation and Certification Server regarding the status of a public key certificate. The document includes a complete example of a time stamping transaction.

Table of Contents

1. Introduction

This document is the result of work that has been proposed and discussed within the IETF PKIX working group. The authors and some members of the group felt that promoting the rather new concepts into the standards process seemed premature. The concepts presented have been stable for some time and partially implemented. It was agreed that a publication as experimental RFC was an appropriate means to

get a stable reference document to permit other implementations to occur.

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [RFC2119].

A Data Validation and Certification Server (DVCS) is a Trusted Third Party (TTP) providing data validation services, asserting correctness of digitally signed documents, validity of public key certificates, and possession or existence of data.

As a result of the validation, a DVCS generates a Data Validation Certificate (DVC). The data validation certificate can be used for constructing evidence of non-repudiation relating to the validity and correctness of an entity's claim to possess data, the validity and revocation status of an entity's public key certificate and the validity and correctness of a digitally signed document.

Services provided by a DVCS do not replace the usage of CRLs and OCSP for public key certificate revocation checking in large open environments, due to concerns about the scalability of the protocol.

It should be rather used to support non-repudiation or to supplement more traditional services concerning paperless document environments. The presence of a data validation certificate supports non-repudiation by providing evidence that a digitally signed document or public key certificate was valid at the time indicated in the DVC.

A DVC validating a public key certificate can for example be used even after the public key certificate expires and its revocation information is no longer or not easily available. Determining the validity of a DVC is assumed to be a simpler task, for example, if the population of DVCS is significantly smaller than the population of public key certificate owners.

An important feature of the protocol is that DVCs can be validated by using the same protocol (not necessarily using the same service), and the validity of a signed document, in particular a DVC, can also be determined by means other than by verifying its signature(s), e.g., by comparing against an archive.

The production of a data validation certificate in response to a signed request for validation of a signed document or public key certificate also provides evidence that due diligence was performed by the requester in validating a digital signature or public key certificate.

This document defines the use of digital signatures to insure the authenticity of documents and DVCs, and uses a corresponding terminology; the use of other methods to provide evidence for authenticity is not excluded, in particular it is possible to replace a SignedData security envelope by another one.

2. Services provided by DVCS

The current specification defines 4 types of validation and certification services:

- Certification of Possession of Data (cpd),
- Certification of Claim of Possession of Data (ccpd),
- Validation of Digitally Signed Document (vsd), and
- Validation of Public Key Certificates (vpkc).

A DVCS MUST support at least a subset of these services. A DVCS may support a restricted vsd service allowing to validate data validation certificates.

On completion of each service, the DVCS produces a data validation certificate - a signed document containing the validation results and trustworthy time information.

2.1 Certification of Possession of Data

The Certification of Possession of Data service provides evidence that the requester possessed data at the time indicated and that the actual data were presented to the Data Validation Server.

2.2 Certification of Claim of Possession of Data

The Certification of Claim of Possession of Data service is similar to the previous one, except that the requester does not present the data itself but a message digest.

2.3 Validation of Digitally Signed Documents

The Validation of Digitally Signed Document service is used when validity of a signed document is to be asserted.

The DVCS verifies all signatures attached to the signed document using all appropriate status information and public key certificates. The DVCS verifies the mathematical correctness of all signatures attached to the document and also checks whether the signing entities can be trusted, for example by validating the full certification path from the signing entities to a trusted point (e.g., the DVCS's CA, or the root CA in a hierarchy).

The DVCS may be able to rely on relevant CRLs or may need to supplement this with access to more current status information from the CAs for example by accessing an OCSP service, a trusted directory service, or other DVCS services.

The DVCS will perform verification of all signatures attached to the signed document. A failure of the verification of one of the signatures does not necessarily result in the failure of the entire validation, and vice versa, a global failure may occur if the document has an insufficient number of signatures.

2.4 Validation of Public Key Certificates

The Validation of Public Key Certificates service is used to verify and assert the validity (according to [RFC2459]) of one or more public key certificates at the specified time.

When verifying a public key certificate, the DVCS verifies that the certificate included in the request is a valid certificate and determines its revocation status at a specified time. DVS checks the full certification path from the certificate's issuer to a trusted point. Again, the DVCS MAY be able to rely on external information (CRL, OCSP, DVCS).

3. Data Certification Server Usage and Scenarii.

It is outside the scope of this document to completely describe different operational scenarii or usages for DVCS.

See Appendix B and C for a set of some basic examples and use cases.

The Validate Signed Document service can be used to support non-repudiation services, to allow use of the signed document beyond public key certificate revocation or expiry, or simply to delegate signature validation to a trusted central (company wide) service.

The Validate Public Key Certificate service can be used when timely information regarding a certificate's revocation status is required (e.g., high value funds transfer or the compromise of a highly sensitive key) or when evidence supporting non-repudiation is required.

A data validation certificate may be used to simplify the validation of a signature beyond the expiry or subsequent revocation of the signing certificate: a Data validation certificate used as an authenticated attribute in a signature includes an additional

assertion about the usability of a certificate that was used for signing. In order to validate such a signature it may be sufficient to only validate the data validation certificate.

A DVCS may include additional key exchange certificates in a data validation certificate to validate a key exchange certificate in order to provide to an application a set of additional authorised recipients for which a session key should also be encrypted. This can be used for example to provide central management of a company wide recovery scheme. Note, that the additional certificates may not only depend on the requested certificate, but also on the requester's identity.

The Certification of Claim of Possession of Data service is also known as time stamping.

The Certification of Possession of Data service can be used to assert legal deposit of documents, or to implement archival services as a trusted third party service.

The Data Validation and Certification Server Protocols can be used in different service contexts. Examples include company-wide centralised services (verification of signatures, certification of company certificates), services to cooperate in a multi-organization community, or general third party services for time stamping or data archival.

An important application of DVCS is an enterprise environment where all security decisions are based on company wide rules. A company wide DVCS service can be used to delegate all technical decisions (e.g., path validation, trust configuration) to a centrally managed service.

In all cases, the trust that PKI entities have in the Data Validation and Certification Server is transferred to the contents of the Data Validation Certificate (just as trust in a CA is transferred to the public key certificates that it issues).

A DVCS service may be combined with or use archiving and logging systems, in order to serve as a strong building block in non-repudiation services. In this sense it can be regarded as an Evidence Recording Authority [ISO-NR].

4. Functional Requirements for DVCS

The DVCS MUST

1. provide a signed response in the form of a data validation certificate to the requester, as defined by policy, or an error response. The DVCS service definition and the policy define how much information that has been used by the DVCS to generate the response will be included in a data validation certificate, e.g., public key certificates, CRLs, and responses from other OCSP servers, DVCS, or others.

2. indicate in the data validation certificate whether or not the signed document, the public key certificate(s), or the data were validated, and, if not, the reason why the verification failed.

3. include a strictly monotonically increasing serial number in each data validation certificate.

4. include a time of day value or a time stamp token into each data validation certificate.

5. sign each data certification token using a key that has been certified with a dvcs signing extended key purpose, and include a reference to this certificate as a signed attribute in the signature.

6. check the validity of its own signing key and certificate before delivering data validation certificates and MUST not deliver data validation certificate in case of failure.

A DVCS SHOULD include within each data validation certificate a policy identifier to determine the trust and validation policy used for DVC's signature.

5. Data Certification Server Transactions

A DVCS transaction begins with a client preparing a Data Validation and Certification Request. The request always contains data for which validity, correctness or possession is to be certified.

The request MAY be encapsulated using a security envelope to provide for authentication of both requester and server. Requester authentication can be achieved by several of the formats described in CMS, in particular, signedData.

The DVCS client chooses an appropriate transport mechanism to convey the requests to a DVCS. It may also be necessary to choose a transport mechanism providing confidentiality and, in particular, allowing authentication of the DVCS by the requestor, e.g., TLS or CMS or S/MIME encryption.

If the request is valid, the DVCS performs all necessary verifications steps, and generates a Data Validation Certificate (DVC), and sends a response message containing the DVC back to the requestor.

The Data Validation Certificate is formed as a signed document (CMS SignedData).

As with the request, it may be necessary to choose a transport mechanism that provides for confidentiality to carry the DVC. DVCs are not necessarily transported the same way as requests, e.g., they can be returned using e-mail after an online request received via HTTPS.

If the request was invalid, the DVCS generates a response message containing an appropriate error notification.

Upon receiving the response, the requesting entity SHOULD verify its validity, i.e., whether it contains an acceptable time, the correct name for the DVCS, the correct request information and message imprint, a valid signature, and satisfactory status, service and policy fields.

When verifying the validity of a DVC, it is up to the requestor's application to check whether a DVCS's signing certificate is valid. Depending on the usage environment, different methods, online or out of band, e.g., CRLs, DVCS, or OCSP, may have to be used.

After all checks have passed, the data validation certificate can be used to authenticate the correctness or possession of the corresponding data.

A DVCS may return more than one DVC corresponding to one request. In this case, all but one request have a global status of 'WAITING'.

6. Identification of the DVCS

In order to be able to import elements from dvcs the following object identifier is used as a ASN.1 module identifier.

```
id-mod-dvcs OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
  dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 15}
```

The DVCS that use SignedData to provide authentication for DVCs MUST sign all data certification messages with a key whose corresponding certificate MUST contain the extended key usage field extension as defined in [RFC2459] Section 4.2.1.14 with KeyPurposeID having value id-kp-dvcs. This extension MUST be marked as critical.

The Data Validation Certificate MUST contain an ESSCertID authenticated attribute for the certificate used by the DVCS for signing.

```
id-kp-dvcs  OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
     dod(6) internet(1) security(5) mechanisms(5) pkix(7) kp(3) 10}
```

Consistent KeyUsage bits:

digitalSignature, nonRepudiation, keyCertSign, cRLSign

A DVCS's certificate MAY contain an Authority Information Access extension [RFC2459] in order to convey the method of contacting the DVCS. The accessMethod field in this extension MUST contain the OID id-ad-dvcs:

```
id-ad-dvcs  OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
     dod(6) internet(1) security(5) mechanisms(5) pkix(7) ad(48) 4}
```

The value of the 'accessLocation' field defines the transport (e.g., an URI) used to access the DVCS.

7. Common Data Types

There are several common data types that occur in the request and the response data structures. These data types are either defined by this document or imported from other sources. This chapter defines and describes these types and lists their usages.

7.1 Version:

The request and the response include an optional integer field specifying the version of the data structure. For both fields the value is 1, or the field is not present at all in this version of the protocol.

7.2 DigestInfo:

This element is defined in [RFC2315]. Since the status of that document is informational, the definition is repeated here:

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm   DigestAlgorithmIdentifier,
    digest            Digest }

Digest ::= OCTET STRING
```

The fields of type DigestInfo have the following meanings:

- The field 'digestAlgorithm' identifies the message-digest algorithm (and any associated parameters) under which data are digested.

- The field 'digest' is the result of the message-digesting process.

A DigestInfo is used in two places:

- as a data portion for the ccpd service, and

- in all a data validation certificates to hold a digest of the data portion of the corresponding request or a copy of the data field for a ccpd service.

7.3. Time Values

Indicators of time can be present in requests and responses. In the most simple form, the time is represented as GeneralizedTime where fractions of seconds are allowed.

An alternate form is a timeStampToken from a TSA, or as a DVC (or some other token) from another third party service.

It is a matter of policy whether a DVCS tries to interpret or validate a Time Value in a request.

```
DVCSTime ::= CHOICE  {
     genTime                    GeneralizedTime,
     timeStampToken             ContentInfo }
```

Future versions of the protocol MAY include additional time formats.

Time values generated by the DVCS are increasing but not necessarily unique, an order among DVCs is defined by serial numbers.

7.4. PKIStatusInfo

This structure is defined in [RFC2510]. It is used as component of the 'chain' field of a TargetEtcChain structure, and as a global status indicator in the DVCSResponse structure. Every occurrence of PKIStatusInfo is generated by the responding DVCS to reflect the result of some local verification.

7.5. TargetEtcChain

A TargetEtcChain structure contains certificates and other indicators to describe either (in a request for a cpkc service) information to be validated, or the result of the verifications. The structure may also contain information about policies and policy mappings.

The details about how to fill in and to interpret the structure are defined later for each service.

The 'pathProcInput' field contains information about policies and policy mapping to be used or used during a validation.

In a response, the 'pkistatus' and 'certstatus' choices can only occur in the 'chain' sequence. If present, they contain the result of a local verification of the immediately preceding element, or of the target value, if it is the first element in the 'chain' sequence. If no 'pkistatus' or 'certstatus' is present, the DVCS considers all elements in the 'chain' as trustworthy. Note, that there may be a valid OCSP response or DVC indicating an invalid certificate.

```
TargetEtcChain ::= SEQUENCE {
     target                      CertEtcToken,
     chain                       SEQUENCE SIZE (1..MAX) OF
                                    CertEtcToken OPTIONAL,
     pathProcInput               [0] PathProcInput OPTIONAL }

PathProcInput ::= SEQUENCE {
     acceptablePolicySet         SEQUENCE SIZE (1..MAX) OF
                                    PolicyInformation,
     inhibitPolicyMapping        BOOLEAN DEFAULT FALSE,
     explicitPolicyReqd          BOOLEAN DEFAULT FALSE }

CertEtcToken ::= CHOICE {

     certificate                 [0] IMPLICIT Certificate ,
     esscertid                   [1] ESSCertId ,
     pkistatus                   [2] IMPLICIT PKIStatusInfo ,
     assertion                   [3] ContentInfo ,
     crl                         [4] IMPLICIT CertificateList,
```

```
        ocspcertstatus              [5] IMPLICIT CertStatus,
        oscpcertid                  [6] IMPLICIT CertId ,
        oscpresponse                [7] IMPLICIT OCSPResponse,
        capabilities                [8] SMIMECapabilities,
        extension                   Extension }
```

Certificate, PolicyInformation and CertificateList are defined in [RFC2459]. ESSCertId is defined in [RFC2634]. CertId, OCSPResponse and CertStatus are defined in [RFC2560]. PKIStatusField is defined in [RFC2510].

The choice 'assertion' can contain a data validation certificate, or a timeStamp, or other assertions.

The choices 'assertion', 'ocspresponse' and 'crl' are provided by services external to the responding DVCS. The choices 'certStatus' and 'pkistatus' reflect decisions made directly by the responding DVCS.

As a replacement for certificates, certification identifiers (ESSCertId, CertId) MAY be used in requests and responses, if this is sufficient to perform the service, e.g., when the corresponding certificates are provided elsewhere in a request or response (as part of the SignedData type).

Certificate or certification identifiers of certification authorities MAY occur in any order and MAY represent several certification chains.

The choice 'capabilities' can be used to indicate SMIMECapabilities. It applies to the certificate identified by the preceding element in the sequence.

7.6. DVCSRequestInformation

A DVCSRequestInformation data structure contains general information about the Data Validation and Certification Request. This structure occurs in a request, and is also included in a corresponding Data Validation Certificate.

```
DVCSRequestInformation ::= SEQUENCE  {

        version                     INTEGER DEFAULT 1 ,
        service                     ServiceType,
        nonce                       INTEGER OPTIONAL,
        requestTime                 DVCSTime OPTIONAL,
        requester                   [0] GeneralNames OPTIONAL,
        requestPolicy               [1] PolicyInformation OPTIONAL,
```

```
        dvcs                        [2] GeneralNames OPTIONAL,
        dataLocations               [3] GeneralNames OPTIONAL,
        extensions                  [4] IMPLICIT Extensions OPTIONAL
}
```

The ServiceType type enumerates the DVCS service type of a request. See chapter 2 for the description of the services.

```
ServiceType ::= ENUMERATED { cpd(1), vsd(2), cpkc(3), ccpd(4) }
```

7.7. GeneralName and GeneralNames

There are several occurrences of SEQUENCES of GeneralName and GeneralNames. These structures are imported from [RFC2459].

8. Data Validation and Certification Requests

A Data Validation and Certification request is a ContentInfo defined in [RFC2630].

It may consist of a [RFC2630] content with a contenttype id-ct-DVCSRequestData signalling a DVCSRequestData,

```
id-ct-DVCSRequestData OBJECT IDENTIFIER ::= {iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) ct(1) 7}
```

These data are optionally encapsulated by contenttypes that provide for authentication and/or confidentiality.

This document describes the usage of a SignedData construct of [RFC2630] where the contenttype indicated in the eContentType of the encapContentInfo is id-ct-DVCSRequestData and the eContent of the encapContentInfo, carried as an octet string, contains a DVCSRequestData structure.

When using a SignedData structure, a Data Validation and Certification Request MAY contain several SignerInfo structures, and countersignature attributes depending on operational environments. When an end user client creates the request, there is one or zero SignerInfo. A relaying DVCS MAY add an additional signature or a countersignature attribute, or MAY use another encapsulation from [RFC2630] that provides for authentication and/or confidentiality.

The content of a request consists of a description of the desired service and additional parameters, the data to be validated, and an optional identifier of the request.

```
DVCSRequest ::= SEQUENCE  {
    requestInformation         DVCSRequestInformation,
    data                       Data,
    transactionIdentifier      GeneralName OPTIONAL
}
```

The 'DVCSRequest.requestInformation' element contains general information about the request. It is filled in by the requester as follows:

- The 'version' field is set to 1 or the field is absent in this version of the protocol.

  The field 'service' contains the requested service.

- The 'nonce' field MAY be used to provide additional protection against replay or content guessing attacks.

- The 'requestTime' field MAY be used to indicate the time for which the requested service should be performed. For a vsd and cpkc service, it specifies the time for which the validity of a signed document or certicates is to be asserted. For the other service, the field is ignored by the DVCS. If the field is absent, the current time is assumed.

- The value of the 'requester' field indicates the requesting entity.

  The interpretation and usage of this field MUST be defined by the DVCS policy.

  Some usage examples are:

  If the field is present, and the request is signed, a DVCS MAY require that the field MUST match the identity (subjectName or subjectAltName extension) of the corresponding signature certificate.

  A request MAY be signed by a DVCS when relaying it to another DVCS.

  When acting as a relay, a DVCS MAY add its own identity in the request relayed to another service provider, and it MAY remove the initial value.

- The 'requestPolicy' field SHOULD indicate the policy under which the validation is requested. This field MUST be checked by the DVCS to verify agreement with its own policy. The absence of this field indicates that any policy is acceptable.

- The 'dvcs' field MAY be used to indicate a list of DVCS which can be contacted to provide (additional) information or to perform additional operations necessary to produce the response.

  It is up to the DVCS policy whether to honor this field or not, and to define which choice of a general name is acceptable (e.g., an URL or a DN).

- The 'dataLocations' field MAY be used to indicate where a copy of the 'data' field of the request or supplementary information can be obtained. The DVCS does not use this field for its own operation, the exact interpretation of this field is defined by applications.

- The 'requestTime' field MAY be used to indicate the time for which the requested service should be performed. For a vsd and cpkc service, it specifies the time for which the validity of a signed document or certicates is to be asserted. For the other service, the field is ignored by the DVCS. If the field is absent, the current time is assumed. The DVCS service may have a time limit or a delta time limit regarding current time which are specified in the local policy of the DVCS service.

- The 'extensions' field MAY be used to include additional information. Extensions may be marked critical or not in order to indicate whether the DVCS is supposed to understand them. This document does not define extensions.

The DVCSRequest.data contains service-specific content, defined by each particular service provided by the DVCS.

Depending on the requested service type, the field may contain a signed document, a list of certificates, a message digest or arbitrary data.

The following type is used:

```
Data ::= CHOICE {
      message           OCTET STRING ,
      messageImprint    DigestInfo,
      certs             SEQUENCE SIZE (1..MAX) OF
                            TargetEtcChain
}
```

The requester fills the 'data' element as follows:

- For a vsd service request, the requestor encapsulates a CMS SignedData object in the value octets of the 'message' choice.

It is up to the requester to decide whether and how to provide any certificate that may be needed to verify the signature(s) in the signedData object. A requester MAY add certificates to the encapsulated signedData object or in the certificate list of the request.

- For a cpkc service request the 'certs' choice is used.

  Each certificate to be verified MUST be included in a separate instance of TargetEtcChain. The 'TargetEtcChain.chain' field, if present, indicates one or more chains of trust that can be used to validate the certificate. The DVCS MAY choose to select a subset of certificates as certification path, or to ignore this field. The 'TargetEtcChain.pathProcInput' field, if present, indicates the acceptable policy set and initial settings for explicit-policy-indicator and inhibit-policy-mapping indicators to be used in X.509 public key certificate path validation (see [RFC2459]).

  Only the Certificate, ESSCertId, CertId or Extension choices of the TargetEtcChain can be used in the request.

  The requester is responsible for providing sufficient information to the DVCS to identify the corresponding certificates.

- For a ccpd service the 'messageImprint' choice is used.

  The hash algorithm indicated in the hashAlgorithm field SHOULD be a "strong" hash algorithm (that is, it SHOULD be one-way and collision resistant). It is up to the Data Certification Server to decide whether or not the given hash algorithm is sufficiently "strong" (based on the current state of knowledge in cryptanalysis and the current state of the art in computational resources, for example).

- For a cpd service the 'message' choice is used.

  The field contains requester-specific data with any type of content. The DVCS does not inspect, modify, or take any particular action based on the particular content of the 'message' field.

The field 'DVCSRequest.transactionIdentifier' MAY be used in order to associate DVCS responses containing error messages, to requests. For example, in a mail based environment, the parameter could be a copy of a messageid. Note, that the transactionIdentifier is not necessary for associating a request with a valid data validation certificate.

9. DVCS Responses

This chapters describes the data structures that are created by a DVCS to indicate the results of validation and certification requests.

A DVCS Response structure is generated by the DVCS as a result of processing of the data validation and certification request.

A Data Validation response contains an [RFC2630] ContentInfo with a type of id-ct-DVCSResponseData signalling a DVCSResponse structure.

```
id-ct-DVCSResponseData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) ct(1) 8 }
```

The data MAY be encapsulated by constructs of [RFC2630] in order to provide authentication of the DVCS, and or integrity and confidentiality of the request. This document specifies the usage of a SignedData construct of [RFC2630].

The contenttype indicated in the eContentType of the encapContentInfo is of type id-ct-DVCSResponseData, signalling a DVCSResponse as eContent of the encapContentInfo (carried as an octet string). The DVCS SHOULD use a key for which a corresponding certificate indicates in an extendedKeyUsage the purpose of DVCS signing.

In a critical situation when a DVCS cannot produce a valid signature (if the DVCS's signing key is known to be compromised, for example), the DVCSResponse, containing the error notification, MUST be generated as a signedData with no signerInfo attached. Receiving unsigned DVCSResponse MUST be treated by the clients as a critical and fatal error, and the content of the message should not be implicitly trusted.

A valid response can contain one of the following:

1. A Data Validation Certificate (DVC), delivering the results of data validation operations, performed by the DVCS.

2. An error notification. This may happen when a request fails due to a parsing error, requester authentication failure, or anything else that prevented the DVCS from executing the request.

The following type is used:

```
DVCSResponse ::= CHOICE {
    dvCertInfo         DVCSCertInfo ,
    dvErrorNote        [0] DVCSErrorNotice }
```

9.1. Data Validation Certificate

A Data Validation Certificate is a signedData object containing a DVCSResponse with a 'dvCertInfo' choice.

```
DVCSCertInfo::= SEQUENCE  {
         version             Integer DEFAULT 1 ,
         dvReqInfo           DVCSRequestInformation,
         messageImprint      DigestInfo,
         serialNumber        Integer,
         responseTime        DVCSTime,
         dvStatus            [0] PKIStatusInfo OPTIONAL,
         policy              [1] PolicyInformation OPTIONAL,
         reqSignature        [2] SignerInfos  OPTIONAL,
         certs               [3] SEQUENCE SIZE (1..MAX) OF
                                 TargetEtcChain OPTIONAL,
         extensions          Extensions OPTIONAL }
```

The DVCSCertInfo structure is returned as a result of successful execution of data validation service. It contains the results of the data validation, a reference to the original request, and other parameters. Please note that 'successful execution' does not necessarily mean that the validation itself was successful - a DVCSCertInfo may contain both the 'valid' and 'invalid' results.

The DVCS creates a DVCSCertInfo as follows:

- The 'version' field is never present in this version of the protocol.

  The 'dvReqInfo' is essentially a copy of the 'requestInformation' field of the corresponding request. The DVCS MAY modify the fields 'dvcs', 'requester', 'dataLocations', and 'nonce' of the ReqInfo structure, e.g., if the request was processed by a chain of DVCS, if the request needs to indicate DVCS, or to indicate where to find a copy of the data from a 'vpd' request. The only modification allowed to a 'nonce' is the inclusion of a new field if it was not present, or to concatenate other data to the end (right) of an existing value.

- The 'DVCSCertInfo.messageImprint' field is computed from the 'data' field of the corresponding request as follows:

  For the 'certs' choice (the 'vpkc' service), the digest is computed over the DER encoded data value. For a 'message' choice (the 'vsd' and the 'vpd' services) the digest is computed over the value octets (not including tag and length octets) of the OCTET STRING. It is up to the DVCS to choose an appropriate digest algorithm.

  For a 'messageImprint' choice (the 'vcpd' service), the 'messageImprint' of the DVCSRequest is copied as is.

- The 'DVCSCertInfo.serialNumber' field contains a unique identifier of the request.

- The field 'responseTime' indicates a time value associated with the response. The value MAY be a locally generated one, or a signed TimeStampToken (TST) or DVC obtained from an external service. Before using a value obtained from an external service, the DVCS must validate it according the rules of the external service.

- The field 'DVCSCertInfo.dvStatus' reflects a collective result of the validation.

  If the field is missing, it is an equivalent of the SUCCESS status.

  For a vkpc, if the status field is present and set to SUCCESS, it indicates that all certificates were successfully validated. If it is present and set to FAILED, it indicates that all or some of the certificates failed validation, and the specific status of the 'certs' should be investigated, at least one of the elements of the 'certs' TargetEtcChain structures MUST have a failure status.

  If the field 'dvStatus' does not indicate success ('granted' or 'granted with mods') the element 'failInfo' MAY indicate the reason for the failure. Note that the field 'certs' MAY contain additional information about verification failures.

  A failure of the verification of one of the signatures does not necessarily result in failing to validate a signed document. For example, as long as a sufficient number of signature was successfully verified, a DVC with status 'grantedWithMods' may be produced. A DVC with status 'granted' MUST only be produced if all signatures verified successfully.

The field MUST be present, and the status must be set to WAITING, if no final response can be immediately available. It is assumed that the DVCS provides an additional final status some time later. The details of the necessary procedures are part of the DVCS policy.

In case of failure, the requester can further investigate the cause of the failure, by looking into the TargetEtcChain fields. 'CertEtctoken.pkistatus' fields will indicate which item(s) has failed or succeeded the validation and for what reason.

- The 'DVCSCertInfo.policy' field indicates the policy under which the DVCS operates.

- If present, 'DVCSCertInfo.reqSignature' MUST be the same value as the signerInfos field of the corresponding request. It is a policy decision whether to include this field.

- The 'DVCSCertInfo.certs' field contains the results of the verifications made by the DVCS. For the cpkc service, each element contains a copy of a corresponding field of the request with the selected subset in the targetAndChain subfield and the results of the verifications, and additional certificates or certificate references, e.g., from certification authorities or as described in appendix C.3. For a vsd service, each element contains the result of the validation of one signature of the signed document to be validated.

  In case of a global status of WAITING, the DVCS MAY choose to return an individual status of waiting in some of the 'certs' field, or not to return such a TargetEtcChain at all.

  The 'acceptablePolicySet' sequence indicates the policies and mappings that were processed during X.509 public key certificate path validation. PolicyMappingsSyntax is defined in [RFC2459].

- The 'extensions' field MAY be used to return additional information to the client. Extensions MAY be marked critical or not in order to indicate whether the client MUST understand them. This document does not define extensions.

9.2. DVCS Error Notification

A DVCS Error Notification is a CMS signedData object containing a DVCSResponse with a 'dvErrorNote' choice.

```
DVCSErrorNotice ::= SEQUENCE {
    transactionStatus          PKIStatusInfo ,
    transactionIdentifier      GeneralName OPTIONAL }
```

The PKIStatusInfo is defined in [RFC2511]. For the purposes of communicating the DVCSErrorNotice, the following subset of PKIFailureInfo values is used:

```
PKIFailureInfo ::= BITSTRING  {

     badRequest       (2),
     -- transaction not permitted or supported
     badTime          (3),
     -- messageTime was not sufficiently close to the system time,
     -- as defined by local policy
     badDataFormat    (5),
     -- the data submitted has the wrong format
     wrongAuthority   (6),
     -- the DVCS indicated in the request is different from the
     -- one creating the response token
     incorrectData    (7)
     --the requester's data (i.e., signature) is incorrect )
```

In the DVCSErrorNotice, the PKIStatus field of the PKIStatusInfo must be set to REJECTED.

The 'statusString' field of PKIStatusInfo can be used to accommodate extra text, such as a reason for the failure, for example "I have gone out of service". The DVCS initializes the 'DVCSErrorNotice.transactionIdentifier' with a copy of the 'DVCSRequest.transactionIdentifier' field of the corresponding request.

In certain circumstances, a DVCS may not be able to produce a valid response to a request (for example, if it is unable to compute signatures for a period of time). In these situations the DVCS MAY create a response with an DVCSErrorNotice but no signature.

DVCS clients SHOULD NOT trust unsigned responses. A DVCS client MAY trust unsigned responses, if the communication channel provides for server authentication (e.g., by services defined by TLS [RFC2246]).

10. Transports

There is no mandatory transport mechanism in this document. All mechanisms are optional. Two examples of transport protocols are given which allow online exchange of request and a response, and asynchronous communication between a client and a DVCS.

A DVCS MAY use a combination of protocols, for example in order to return additional DVCs.

10.1 DVCS Protocol via HTTP or HTTPS

This subsection specifies a means for conveying ASN.1-encoded messages for the DVCS protocol exchanges via the HyperText Transfer Protocol.

The DER encoded DVCS requests and responses are encapsulated using a simple MIME object with Content-Type application/dvcs (and with the default binary encoding).

This MIME object can be sent and received using common HTTP or HTTPS processing engines over WWW links and provides a simple client-server transport for DVCS messages.

10.2 DVCS Protocol Using Email

This section specifies a means for conveying ASN.1-encoded messages for the protocol exchanges described in Section 8 via Internet mail.

The DER encoded DVCS requests and responses are encapsulated using a simple MIME object with Content-Type application/dvcs with an appropriate Content-Transfer-Encoding.

This MIME object can be sent and received using MIME processing engines and provides a simple Internet mail transport for DVCS messages.

In order to be able to associate a possible error response with a request, the requester SHOULD use the field 'transactionIdentifier'. The requester SHOULD not make any assumption about the usage of message header fields by the responding service, in particular the usage of fields like Subject, Message-ID or References.

11. Security Considerations

This entire chapter discusses security considerations.

When designing a data validation and certification service, the following considerations have been identified that have an impact upon the validity or "trust" in the data validation certificate.

It is imperative that keys used to sign DVCs are guarded with proper security and controls in order to minimize the possibility of compromise. Nevertheless, in case the private key does become compromised, an audit trail of all the DVC generated by the DVCS SHOULD be kept as a means to help discriminate between genuine and false DVCs. A DVCS MAY provide for a vsd service to validate DVCs created by this DVCS or another one solely based on the audit trail.

When confidentiality and server authentication is required, requests and responses MAY be protected using appropriate mechanisms (e.g., CMS encapsulation [RFC 2630] or TLS [RFC2246]).

Server authentication is highly recommended for the vsd and cpd service.

Client identification and authentication MAY use services defined by TLS [RFC2246]) instead of, or in addition to, using a CMS format providing authentication.

12. Patent Information

The following United States Patents related to data validation and certification services, listed in chronological order, are known by the authors to exist at this time. This may not be an exhaustive list. Other patents may exist or be issued at any time. Implementers of the DVCS protocol and applications using the protocol SHOULD perform their own patent search and determine whether or not any encumberences exist on their implementation.

4,309,569 Method of Providing Digital Signatures
(issued) January 5, 1982
(inventor) Ralph C. Merkle
(assignee) The Board of Trustees of the Leland Stanford Junior University

5,001,752 Public/Key Date-Time Notary Facility
(issued) March 19, 1991
(inventor) Addison M. Fischer

5,022,080 Electronic Notary
(issued) June 4, 1991
(inventors) Robert T. Durst, Kevin D. Hunter

5,136,643 Public/Key Date-Time Notary Facility
(issued) August 4, 1992
(inventor) Addison M. Fischer
(Note: This is a continuation of patent # 5,001,752.)

5,136,646 Digital Document Time-Stamping with Catenate Certificate
(issued) August 4, 1992
(inventors) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Bell Communications Research, Inc.,

5,136,647 Method for Secure Time-Stamping of Digital Documents
(issued) August 4, 1992
(inventors) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Bell Communications Research, Inc.,

5,373,561 Method of Extending the Validity of a Cryptographic Certificate
(issued) December 13, 1994
(inventors) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Bell Communications Research, Inc.,

5,422,95 Personal Date/Time Notary Device
(issued) June 6, 1995
(inventor) Addison M. Fischer

5,781,629 Digital Document Authentication System
(issued) July 14, 1998
(inventor) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Surety Technologies, Inc.,

13. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2510] Adams, C. and S. Farrell, "Internet X.509 Public Key Infrastructure, Certificate Management Protocols", RFC 2510, March 1999.

[RFC2459] Housley, R., Ford, W., Polk, W. and D. Solo, "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile", RFC 2459, January 1999.

[RFC2630] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[ISONR] ISO/IEC 10181-5: Security Frameworks in Open Systems. Non-Repudiation Framework.

[RFC2119] Bradner, S., "Key works for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2511] Myers, M., Adams, C., Solo, D. and D. Kemp, "Internet X.509 Certificate Request Message Format", RFC 2511, March 1999.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol, Version 1.0", RFC 2246, January 1999.

[RFC2634] Hoffman P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S. and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol", RFC 2560, June 1999.

14. Authors' Addresses

Carlisle Adams
Entrust Technologies
1000 Innovation Drive
Ottawa, Ontario
K2K 3E7
CANADA

EMail: cadams@entrust.com

Michael Zolotarev
Baltimore Technologies Pty Limited
5th Floor, 1 James Place
North Sydney, NSW 2060
AUSTRALIA

EMail: mzolotarev@baltimore.com

Peter Sylvester
EdelWeb SA - Groupe ON-X Consulting
15, Quai de Dion Bouton
F-92816 Puteaux Cedex
FRANCE

EMail: peter.sylvester@edelweb.fr

Robert Zuccherato
Entrust Technologies
1000 Innovation Drive
Ottawa, Ontario
K2K 3E7
CANADA

EMail: robert.zuccherato@entrust.com

APPENDIX A - PKCS #9 Attribute

We define a PKCS #9 [PKCS9] attribute type. The attribute type has ASN.1 type SignedData and contains a data validation certificate.

The object identifier id-aa-dvcs-dvc identifies the data validation certificate attribute type.

```
id-aa-dvcs-dvc OBJECT IDENTIFIER ::= {iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) aa(2) 29}
```

The attribute may be used as an authenticated or unauthenticated attribute in CMS SignedData documents.

APPENDIX B - Signed document validation.

We present some examples of a possible use of DVCS in the context of validation of signed documents.

B.1 Signed document validation

The example covers the case where a DVCS is used by a signer to obtain a proof that a document’s structure, including one or more attached signatures, is/was correct, after the document was signed.

The DVC can be produced either by a DVCS that is trusted by the signer, or by a DVCS that is trusted by an intended verifier of the document.

The signer uses the obtained DVC as an evidence that its intentions were good and it produced a signed document using the environment(keys, algorithms, etc) that was known to be OK.

It produces a stand-alone document that can be used to extend the life of a signature. This example assumes that we have total trust in the Data Validation and Certification Server.

Signature algorithms and keys have a finite lifetime. Therefore, signatures have a finite lifetime. The Data Certification Server can be used to extend the lifetime of a signature.

In order to extend the lifetime of a signature in this way, the following technique can be used:

1) The signature needs to be certified:

   The signed message is presented to the Data Validation and Certification Server in a ’vsd’ service request.

The DVCS verifies that the signature and certificates are valid at that time by checking expiry dates, status information, or DVCs, and returns a DVC.

2) The DVC SHOULD be verified.

The signature of the Data Validation and Certification Server in data certification token SHALL be verified using the Data Certification Server's valid verification key.

A signer's signing key (and therefore, its signature) is only valid until some specified time T1. The DVCS's signing key (and therefore, its signature) is valid until some specified time T2 that is (usually) after time T1. Without certification, the signer's signature would only be valid until time T1. With certification, the signer's signature remains valid until time T2, regardless of subsequent revocation or expiry at time T1.

If the signature of the DVCS is valid, the trust we have in the DVCS allows us to conclude that the original signature on the data was valid at the time included in the DVC.

The DVCS signing key MUST be of a sufficient length to allow for a sufficiently long lifetime. Even if this is done, the key will have a finite lifetime. Since data validation certificates are just another type of signed documents, they can be validated using (another) DVCS.

APPENDIX C - Verifying the Status of a Public Key Certificate

We now present three examples of how to produce a data validation certificate that can be used to assert that a public key certificate is valid, trusted, and can be used for a particular purpose.

A client wants to use a given public key certificate either to use it to verify a signature on a document or to use it for document encryption.

A DVCS MUST have access to current information regarding public certificate status, it can therefore be used to verify the revocation status of a certificate at the current time.

The following technique can be used:

A) The public key certificate needs to be validated.

The certificate is presented to the Data Certification Server using a 'vpkc' service.

The Data Validation and Certification Server verifies that the public key certificate is valid and that it hasn't been revoked and then returns a data validation certificate.

B) The data validation certificate MUST be verified.

The signature of the Data Certification Server in the data certification token SHALL be verified using the Data Validation and Certification Server's valid certificate.

C) The public key certificate is used:

C.1) A clients's own public key certificate (i.e., the corresponding private key) can be used to add a signature to a document. The signing certificate and the data validation certificate can be added as signed attributes to the signature.

A data validation certificate can now be used during the validation signatures using the key contained in the public key certificate. This service provided by the DVCS can be thought of as a supplement to the usual method of checking revocation status.

In other words, signature validation at a later time does not necessarily require access to the revocation status of the user's signing certificate, access to a DVCS service and validation of the DVC is sufficient to verify a signature. Note that the DVC does not tell when the signature had been created, it only indicates when the signing certificate was valid.

C.2) A public key certificate for key exchange can be used after having obtained a data validation certification certificate to encrypt data. The DVC can be stored with the data and/or stored by the creator of the encrypted document.

If an intended recipient of the document claims that the creator did not use an appropriate encryption key, the DVC (obtained by a recipient's DVCS) can be used as evidence that the recipient's DVCS has authorized the usage of the public key.

C.3) The procedure described in the previous paragraph can be enhanced to provide domain encryption in several ways. Organizations require that encrypted documents need to be recoverable. One simple way is to always encrypt documents with additional recipients that act as 'domain encryption centers' or 'recovery centers'. This is not a technically difficult

problem, but may require complicated and difficult interactions with the end user, in particular when the document's recipients are in several different organizations.

One possible solution consists of adding additional certificates to the dvc that validates the usage of a particular public key certificate used for encryption. In an environment of several organizations, one of the possible procedures may be:

The client asks its local dvcs to validate the public key certificate. The dvcs forwards the request to a dvcs of a remote organization. The remotes organization's dvcs verifies the certificate and provides a dvc assertion validating the certificate. It adds additional certificates usable for key exchange to the certEtcChain structure indicating additional required recipients. The local dvc creates a dvc containing the dvc of the remote dvcs. It may add additional certificates or references to the dvc. The clients use all validated certificates to be usable for key exchange to enhance its list of recipients.

In the local dvcs may as well use local information about the remote organization's need for additional recipients.

Appendix D - MIME Registration

To: ietf-types@iana.org Subject: Registration of MIME media type application/timestamp

MIME media type name: application

MIME subtype name: dvcs

Required parameters: None

Optional parameters: None

Encoding considerations: binary or Base64

Security considerations: Carries a request for a data validation and certification service and the response. A request may be cryptographically signed. The response will be cryptographically signed.

Interoperability considerations: None

Published specification:
RFC 3029 on Data Validation and Certification Server Protocols

Applications which use this media type: Data Validation and Certification Servers and Clients

Additional information:

Magic number(s): None
File extension(s): .dvc
Macintosh File Type Code(s): none

Person & email address to contact for further information: Peter Sylvester <peter.sylvester@edelweb.fr>

Intended usage: COMMON

Author/Change controller: Peter Sylvester <peter.sylvester@edelweb.fr>

Appendix E - ASN.1 Module using 1988 Syntax

```
PKIXDVCS {iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-dvcs(15)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS
  Extensions, AlgorithmIdentifier
  FROM PKIX1Explicit88 {iso(1) identified-organization(3)
  dod(6) internet(1) security(5) mechanisms(5) pkix(7)
  id-mod(0) id-pkix1-explicit-88(1)}

  GeneralName, PolicyInformation
  FROM PKIX1Implicit88 {iso(1) identified-organization(3)
  dod(6) internet(1) security(5) mechanisms(5) pkix(7)
  id-mod(0) id-pkix1-implicit-88(2)}

  PKIStatusInfo, PKIStatusField FROM PKIXCMP {iso(1)
  identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) id-mod(0)
  id-mod-cmp(9)}

  ContentInfo FROM CryptographicMessageSyntax {iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) modules(0) cms(1)}
```

```
  ESSCertID FROM ExtendedSecurityServices
  { iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs-9(9) smime(16) modules(0) ess(2) }

  CertId, OCSPResponse, CertStatus
  FROM OCSP
  {iso(1) identified-organization(3)
  dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-ocsp(14)}

  SMIMECapabilities FROM SecureMimeMessageV3
  { iso(1) member-body(2) us(840) rsadsi(113549)
   pkcs(1) pkcs-9(9) smime(16) modules(0) smime(4) }

  ;

-- Authority Information Access for DVCS

id-ad-dvcs  OBJECT IDENTIFIER ::= {id-pkix id-ad(48) 4}

-- Key Purpose for DVCS

id-kp-dvcs  OBJECT IDENTIFIER ::= {id-pkix id-kp(3) 10}

-- eContentType for a dvcs requests and responses

id-ct-DVCSRequestData  OBJECT IDENTIFIER ::= { id-smime ct(1) 7 }
id-ct-DVCSResponseData OBJECT IDENTIFIER ::= { id-smime ct(1) 8 }

-- Data validation certificate attribute

id-aa-dvcs-dvc OBJECT IDENTIFIER ::= { id-smime aa(2) 29 }

-- using the following bases :

id-pkix     OBJECT IDENTIFIER ::= {iso(1)
               identified-organization(3) dod(6)
               internet(1) security(5) mechanisms(5) pkix(7)}

id-smime    OBJECT IDENTIFIER ::= { iso(1) member-body(2)
               us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 16 }

Version ::= Integer

DigestInfo ::= SEQUENCE {
    digestAlgorithm   DigestAlgorithmIdentifier,
    digest            Digest
}
```

```
Digest ::= OCTET STRING

Nonce ::= Integer

DVCSTime ::= CHOICE  {
     genTime                     GeneralizedTime,
     timeStampToken              ContentInfo
}
TargetEtcChain ::= SEQUENCE {
     target                      CertEtcToken,
     chain                       SEQUENCE SIZE (1..MAX) OF
                                    CertEtcToken OPTIONAL,
     pathProcInput               [0] PathProcInput OPTIONAL
}

PathProcInput ::= SEQUENCE {
     acceptablePolicySet         SEQUENCE SIZE (1..MAX) OF
                                    PolicyInformation,
     inhibitPolicyMapping        BOOLEAN DEFAULT FALSE,
     explicitPolicyReqd          BOOLEAN DEFAULT FALSE
}

CertEtcToken ::= CHOICE {
     certificate                 [0] IMPLICIT Certificate ,
     esscertid                   [1] ESSCertId ,
     pkistatus                   [2] IMPLICIT PKIStatusInfo ,
     assertion                   [3] ContentInfo ,
     crl                         [4] IMPLICIT CertificateList,
     ocspcertstatus              [5] IMPLICIT CertStatus,
     oscpcertid                  [6] IMPLICIT CertId ,
     oscpresponse                [7] IMPLICIT OCSPResponse,
     capabilities                [8] SMIMECapabilities,
     extension                   Extension
}

DVCSRequestInformation ::= SEQUENCE  {
        version                     INTEGER DEFAULT 1 ,
        service                     ServiceType,
        nonce                       Nonce OPTIONAL,
        requestTime                 DVCSTime OPTIONAL,
        requester                   [0] GeneralNames OPTIONAL,
        requestPolicy               [1] PolicyInformation OPTIONAL,
        dvcs                        [2] GeneralNames OPTIONAL,
        dataLocations               [3] GeneralNames OPTIONAL,
        extensions                  [4] IMPLICIT Extensions OPTIONAL
}

ServiceType ::= ENUMERATED { cpd(1), vsd(2), cpkc(3), ccpd(4) }
```

```
DVCSRequest ::= SEQUENCE  {
    requestInformation          DVCSRequestInformation,
    data                        Data,
    transactionIdentifier       GeneralName OPTIONAL
}

Data ::= CHOICE {
      message           OCTET STRING ,
      messageImprint    DigestInfo,
      certs             SEQUENCE SIZE (1..MAX) OF
                            TargetEtcChain
}

DVCSResponse ::= CHOICE
{
    dvCertInfo         DVCSCertInfo ,
    dvErrorNote        [0] DVCSErrorNotice
}

DVCSCertInfo::= SEQUENCE  {
         version              Integer DEFAULT 1 ,
         dvReqInfo            DVCSRequestInformation,
         messageImprint       DigestInfo,
         serialNumber         Integer,
         responseTime         DVCSTime,
         dvStatus             [0] PKIStatusInfo OPTIONAL,
         policy               [1] PolicyInformation OPTIONAL,
         reqSignature         [2] SignerInfos  OPTIONAL,
         certs                [3] SEQUENCE SIZE (1..MAX) OF
                                  TargetEtcChain OPTIONAL,
         extensions           Extensions OPTIONAL
}

DVCSErrorNotice ::= SEQUENCE {
    transactionStatus           PKIStatusInfo ,
    transactionIdentifier       GeneralName OPTIONAL
}

END
```

Appendix F - Examples

This chapter contains an example of a request and a response of a 'Certify Claim of Possession of Data' transaction of the Clepsydre Demonstration Project sponsored by La Poste, France.

The information has been formatted with a slightly modified version of Peter Gutmann's dumpasn1 program.

The response Data Validation Certificate contains the signing certificate.

The data that are time stamped is the binary of the client program used to make the request.

Request:

```
 0 30  582: SEQUENCE {
 4 06    9:  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
           : . (PKCS #7)
15 A0  567:  [0] {
19 30  563: . SEQUENCE {
23 02    1: .  INTEGER 3
26 31   11: .  SET {
28 30    9: . . SEQUENCE {
30 06    5: . .  OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
37 05    0: . .  NULL
           : . .  }
           : . . }
39 30  153: .  SEQUENCE {
42 06   11: . . OBJECT IDENTIFIER
           : . .  id-ct-DVCSRequestData (1 2 840 113549 1 9 16 1 7)
           : . .  (S/MIME Content Types (1 2 840 113549 1 9 16 1))
55 A0  137: . . [0] {
58 04  134: . .  OCTET STRING, encapsulates {
61 30  131: . . .  SEQUENCE {
64 30   96: . . . . SEQUENCE {
66 0A    1: . . . .  ENUMERATED CCPD (4)
69 A0   77: . . . .  [0] {
71 A4   75: . . . . . [4] {
73 30   73: . . . . .  SEQUENCE {
75 31   11: . . . . . . SET {
77 30    9: . . . . . .  SEQUENCE {
79 06    3: . . . . . . . OBJECT IDENTIFIER
           : . . . . . . .  countryName (2 5 4 6)
           : . . . . . . .  (X.520 id-at (2 5 4))
84 13    2: . . . . . . . PrintableString 'FR'
           : . . . . . . . }
           : . . . . . .  }
88 31   14: . . . . . . SET {
90 30   12: . . . . . .  SEQUENCE {
92 06    3: . . . . . . . OBJECT IDENTIFIER
           : . . . . . . .  localityName (2 5 4 7)
           : . . . . . . .  (X.520 id-at (2 5 4))
97 13    5: . . . . . . . PrintableString 'Paris'
           : . . . . . . . }
           : . . . . . .  }
```

```
 104 31   16: . . . . . . SET {
 106 30   14: . . . . . .  SEQUENCE {
 108 06    3: . . . . . .  OBJECT IDENTIFIER
             : . . . . . .   organizationName (2 5 4 10)
             : . . . . . .   (X.520 id-at (2 5 4))
 113 13    7: . . . . . .  PrintableString 'EdelWeb'
             : . . . . . . }
             : . . . . . .  }
 122 31   24: . . . . . . SET {
 124 30   22: . . . . . .  SEQUENCE {
 126 06    3: . . . . . .  OBJECT IDENTIFIER
             : . . . . . .   commonName (2 5 4 3)
             : . . . . . .   (X.520 id-at (2 5 4))
 131 13   15: . . . . . .  PrintableString 'Peter Sylvester'
             : . . . . . . }
             : . . . . . .  }
             : . . . . . . }
             : . . . . . .  }
             : . . . . . . }
 148 A1   12: . . . .  [1] {
 150 06   10: . . . . . OBJECT IDENTIFIER '1 3 6 1 4 1 5309 1 2 1'
             : . . . . . }
             : . . . . .  }
 162 30   31: . . . . SEQUENCE {
 164 30    7: . . . .  SEQUENCE {
 166 06    5: . . . . . OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
             : . . . . .  (OIW)
             : . . . . . }
 173 04   20: . . . .  OCTET STRING
             : . . . .  75 B6 85 AF 6F 89 46 7D E8 07 15 25 1E 45 97 8F
             : . . . .  CD 1F A5 66
             : . . . .  }
             : . . . . }
             : . . .  }
             : . . . }
 195 31  387: .  SET {
 199 30  383: . . SEQUENCE {
 203 02    1: . .  INTEGER 1
 206 30  124: . .  SEQUENCE {
 208 30  112: . . . SEQUENCE {
 210 31   11: . . .  SET {
 212 30    9: . . . . SEQUENCE {
 214 06    3: . . . .  OBJECT IDENTIFIER countryName (2 5 4 6)
             : . . . . . (X.520 id-at (2 5 4))
 219 13    2: . . . .  PrintableString 'FR'
             : . . . .  }
             : . . . . }
```

```
 223 31   21: . . .  SET {
 225 30   19: . . . . SEQUENCE {
 227 06    3: . . . .  OBJECT IDENTIFIER organizationName (2 5 4 10)
             : . . . . . (X.520 id-at (2 5 4))
 232 13   12: . . . .  PrintableString 'EdelWeb S.A.'
             : . . . .  }
             : . . . . }
 246 31   40: . . .  SET {
 248 30   38: . . . . SEQUENCE {
 250 06    3: . . . .  OBJECT IDENTIFIER
             : . . . . . organizationalUnitName (2 5 4 11)
             : . . . . . (X.520 id-at (2 5 4))
 255 13 31: . . . .  PrintableString 'Clepsydre Demonstration Service'
             : . . . .  }
             : . . . . }
 288 31   32: . . .  SET {
 290 30   30: . . . . SEQUENCE {
 292 06    3: . . . .  OBJECT IDENTIFIER commonName (2 5 4 3)
             : . . . . . (X.520 id-at (2 5 4))
 297 13   23: . . . .  PrintableString 'Time Stamping Authority'
             : . . . .  }
             : . . . . }
             : . . .  }
 322 02    8: . . . INTEGER
             : . . .  00 94 88 17 21 34 37 76
             : . . . }
 332 30    9: . .  SEQUENCE {
 334 06    5: . . . OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
             : . . .  (OIW)
 341 05    0: . . . NULL
             : . . . }
 343 A0   95: . .  [0] {
 345 30   26: . . . SEQUENCE {
 347 06    9: . . .  OBJECT IDENTIFIER
             : . . . . contentType (1 2 840 113549 1 9 3)
             : . . . . (PKCS #9 (1 2 840 113549 1 9))
 358 31   13: . . .  SET {
 360 06   11: . . . . OBJECT IDENTIFIER
             : . . . .  id-ct-dvcsrequest (1 2 840 113549 1 9 16 1 7)
             : . . . .  (S/MIME Content Types (1 2 840 113549 1 9 16 1))
             : . . . . }
             : . . .  }
 373 30   28: . . . SEQUENCE {
 375 06    9: . . .  OBJECT IDENTIFIER
             : . . . . signingTime (1 2 840 113549 1 9 5)
             : . . . . (PKCS #9 (1 2 840 113549 1 9))
 386 31   15: . . .  SET {
 388 17   13: . . . . UTCTime '000417171457Z'
```

```
            : . . . . }
            : . . .  }
 403 30   35: . . . SEQUENCE {
 405 06    9: . . .  OBJECT IDENTIFIER
            : . . . . messageDigest (1 2 840 113549 1 9 4)
            : . . . . (PKCS #9 (1 2 840 113549 1 9))
 416 31   22: . . .  SET {
 418 04   20: . . . . OCTET STRING
            : . . . .  4D A8 C2 D2 CE 7C 0D 04 41 2F 44 13 33 75 DB 2F
            : . . . .  5B 2D F9 DC
            : . . . . }
            : . . .  }
            : . . . }
 440 30   13: . .  SEQUENCE {
 442 06    9: . . . OBJECT IDENTIFIER
            : . . .  rsaEncryption (1 2 840 113549 1 1 1)
            : . . .  (PKCS #1)
 453 05    0: . . . NULL
            : . . . }
 455 04  128: . .  OCTET STRING
            : . . . 6E 7B 0E 36 F5 08 5F 16 3C 31 7B 28 BB 0B C2 C6
            : . . . 17 67 A6 B5 54 F1 98 E2 6F 89 96 0E 0C 99 E6 CB
            : . . . 40 C1 9B 8D D8 D7 8E D3 2B 41 F7 16 26 5B B7 08
            : . . . BF E6 95 B2 D9 01 6C FE B1 2C 52 C1 5A D2 31 F3
            : . . . 8E CA DD 11 A1 72 05 29 41 6A DD 28 40 AA 5C 77
            : . . . C6 9D 1D 80 53 DB 6F 9C 4C A5 A3 8F 92 8B 18 3F
            : . . . D5 3A AD 01 87 69 C3 FD D3 D8 C3 D0 CA 6B E6 0D
            : . . . 4E 53 6E 50 20 99 7C 94 C2 44 25 1B 06 C0 99 96
            : . .  }
            : . . }
            : .  }
            : . }
            :  }
```

The corresponding data in PEM format are:

```
-----BEGIN PKCS7-----
MIICRgYJKoZIhvcNAQcCoIICNzCCAjMCAQMxCzAJBgUrDgMCGgUAMIGZBgsqhkiG
9w0BCRABB6CBiQSBhjCBgzBgCgEEoE2kSzBJMQswCQYDVQQGEwJGUjEOMAwGA1UE
BxMFUGFyaXMxEDAOBgNVBAoTB0VkZWxXZWIxGDAWBgNVBAMTD1BldGVyIFN5bHZl
c3RlcqEMBgorBgEEAak9AQIBMB8wBwYFKw4DAhoEFHW2ha9viUZ96AcVJR5Fl4/N
H6VmMYIBgzCCAX8CAQEwfDBwMQswCQYDVQQGEwJGUjEVMBMGA1UEChMMRWRlbFdl
YiBTLkEuMSgwJgYDVQQLEx9DbGVwc3lkcmUgRGVtb25zdHJhdGlvbiBTZXJ2aWNl
MSAwHgYDVQQDExdUaW1lIFN0YW1waW5nIEF1dGhvcml0eQIIAJSIFyE0N3YwCQYF
Kw4DAhoFAKBfMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0BCRABBzAcBgkqhkiG9w0B
CQUxDxcNMDAwNDE3MTcxNDU3WjAjBgkqhkiG9w0BCQQxFgQUTajC0s58DQRBL0QT
M3XbL1st+dwwDQYJKoZIhvcNAQEBBQAEgYBuew429QhfFjwxeyi7C8LGF2emtVTx
mOJviZYODJnmy0DBm43Y147TK0H3FiZbtwi/5pWy2QFs/rEsUsFa0jHzjsrdEaFy
```

```
BSlBat0oQKpcd8adHYBT22+cTKWjj5KLGD/VOq0Bh2nD/dPYw9DKa+YNTlNuUCCZ
fJTCRCUbBsCZlg==
-----END PKCS7-----

Response:

   0 30 2039: SEQUENCE {
   4 06    9:  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
             : . (PKCS #7)
  15 A0 2024:  [0] {
  19 30 2020: . SEQUENCE {
  23 02    1: .  INTEGER 3
  26 31   11: .  SET {
  28 30    9: . . SEQUENCE {
  30 06    5: . .  OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
             : . . . (OIW)
  37 05    0: . .  NULL
             : . .  }
             : . . }
  39 30  301: .  SEQUENCE {
  43 06   11: . . OBJECT IDENTIFIER
             : . .  id-ct-DVCSResponseData (1 2 840 113549 1 9 16 1 8)
             : . .  (S/MIME Content Types (1 2 840 113549 1 9 16 1))
  56 A0  284: . . [0] {
  60 04  280: . .  OCTET STRING, encapsulates {
  64 30  276: . . .  SEQUENCE {
  68 30  214: . . . . SEQUENCE {
  71 0A    1: . . . .  ENUMERATED CCPD (4)
  74 A0   77: . . . .  [0] {
  76 A4   75: . . . . . [4] {
  78 30   73: . . . . .  SEQUENCE {
  80 31   11: . . . . . . SET {
  82 30    9: . . . . . .  SEQUENCE {
  84 06    3: . . . . . . . OBJECT IDENTIFIER
             : . . . . . . .  countryName (2 5 4 6)
             : . . . . . . .  (X.520 id-at (2 5 4))
  89 13    2: . . . . . . . PrintableString 'FR'
             : . . . . . . . }
             : . . . . . .  }
  93 31   14: . . . . . . SET {
  95 30   12: . . . . . .  SEQUENCE {
  97 06    3: . . . . . . . OBJECT IDENTIFIER
             : . . . . . . .  localityName (2 5 4 7)
             : . . . . . . .  (X.520 id-at (2 5 4))
 102 13    5: . . . . . . . PrintableString 'Paris'
             : . . . . . . . }
             : . . . . . .  }
 109 31   16: . . . . . . SET {
```

```
 111 30   14: . . . . . .  SEQUENCE {
 113 06    3: . . . . . . . OBJECT IDENTIFIER
              : . . . . . .  organizationName (2 5 4 10)
              : . . . . . .  (X.520 id-at (2 5 4))
 118 13    7: . . . . . . . PrintableString 'EdelWeb'
              : . . . . . . . }
              : . . . . . .  }
 127 31   24: . . . . . . SET {
 129 30   22: . . . . . .  SEQUENCE {
 131 06    3: . . . . . . . OBJECT IDENTIFIER
              : . . . . . .  commonName (2 5 4 3)
              : . . . . . .  (X.520 id-at (2 5 4))
 136 13   15: . . . . . . . PrintableString 'Peter Sylvester'
              : . . . . . . . }
              : . . . . . .  }
              : . . . . . . }
              : . . . . . .  }
              : . . . . . . }
 153 A1   12: . . . .  [1] {
 155 06   10: . . . . . OBJECT IDENTIFIER '1 3 6 1 4 1 5309 1 2 1'
              : . . . . . }
 167 A2  116: . . . .  [2] {
 169 A4  114: . . . . . [4] {
 171 30  112: . . . . .  SEQUENCE {
 173 31   11: . . . . . . SET {
 175 30    9: . . . . . .  SEQUENCE {
 177 06    3: . . . . . . . OBJECT IDENTIFIER
              : . . . . . .  countryName (2 5 4 6)
              : . . . . . .  (X.520 id-at (2 5 4))
 182 13    2: . . . . . . . PrintableString 'FR'
              : . . . . . . . }
              : . . . . . .  }
 186 31   21: . . . . . . SET {
 188 30   19: . . . . . .  SEQUENCE {
 190 06    3: . . . . . . . OBJECT IDENTIFIER
              : . . . . . .  organizationName (2 5 4 10)
              : . . . . . .  (X.520 id-at (2 5 4))
 195 13   12: . . . . . . . PrintableString 'EdelWeb S.A.'
              : . . . . . . . }
              : . . . . . .  }
 209 31   40: . . . . . . SET {
 211 30   38: . . . . . .  SEQUENCE {
 213 06    3: . . . . . . . OBJECT IDENTIFIER
              : . . . . . .  organizationalUnitName (2 5 4 11)
              : . . . . . .  (X.520 id-at (2 5 4))
 218 13 31: . . . . . PrintableString 'Clepsydre Demonstration Service'
              : . . . . . . . }
              : . . . . . .  }
```

```
 251 31   32: . . . . . . SET {
 253 30   30: . . . . . .  SEQUENCE {
 255 06    3: . . . . . . OBJECT IDENTIFIER
             : . . . . . .  commonName (2 5 4 3)
             : . . . . . .  (X.520 id-at (2 5 4))
 260 13   23: . . . . . . PrintableString 'Time Stamping Authority'
             : . . . . . . }
             : . . . . . .  }
             : . . . . . . }
             : . . . . .  }
             : . . . . . . }
             : . . . . .  }
 285 30   31: . . . . SEQUENCE {
 287 30    7: . . . .  SEQUENCE {
 289 06    5: . . . . . OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
             : . . . . . }
 296 04   20: . . . .  OCTET STRING
             : . . . .  75 B6 85 AF 6F 89 46 7D E8 07 15 25 1E 45 97 8F
             : . . . .  CD 1F A5 66
             : . . . .  }
 318 02    7: . . . . INTEGER
             : . . . .  01 78 0A 1E CA 88 23
 327 18   15: . . . . GeneralizedTime '20000417171617Z'
             : . . . . }
             : . . .  }
             : . .  }
             : . . }
 344 A0  992: .  [0] {
 348 30  988: . . SEQUENCE {
 352 30  708: . .  SEQUENCE {
 356 A0    3: . . . [0] {
 358 02    1: . . .  INTEGER 2
             : . . .  }
 361 02    8: . . . INTEGER
             : . . .  00 94 88 17 17 64 37 32
 371 30   13: . . . SEQUENCE {
 373 06    9: . . .  OBJECT IDENTIFIER
             : . . . . md5withRSAEncryption (1 2 840 113549 1 1 4)
             : . . . . (PKCS #1)
 384 05    0: . . .  NULL
             : . . .  }
 386 30  112: . . . SEQUENCE {
 388 31   11: . . .  SET {
 390 30    9: . . . . SEQUENCE {
 392 06    3: . . . .  OBJECT IDENTIFIER countryName (2 5 4 6)
             : . . . . . (X.520 id-at (2 5 4))
 397 13    2: . . . .  PrintableString 'FR'
             : . . . .  }
```

```
           : . . . . }
 401 31   21: . . .  SET {
 403 30   19: . . . . SEQUENCE {
 405 06    3: . . . .  OBJECT IDENTIFIER organizationName (2 5 4 10)
           : . . . . . (X.520 id-at (2 5 4))
 410 13   12: . . . .  PrintableString 'EdelWeb S.A.'
           : . . . .  }
           : . . . . }
 424 31   40: . . .  SET {
 426 30   38: . . . . SEQUENCE {
 428 06    3: . . . .  OBJECT IDENTIFIER
           : . . . . . organizationalUnitName (2 5 4 11)
           : . . . . . (X.520 id-at (2 5 4))
 433 13 31: . . . .  PrintableString 'Clepsydre Demonstration Service'
           : . . . .  }
           : . . . . }
 466 31   32: . . .  SET {
 468 30   30: . . . . SEQUENCE {
 470 06    3: . . . .  OBJECT IDENTIFIER commonName (2 5 4 3)
           : . . . . . (X.520 id-at (2 5 4))
 475 13   23: . . . .  PrintableString 'Time Stamping Authority'
           : . . . .  }
           : . . . . }
           : . . .  }
 500 30   30: . . . SEQUENCE {
 502 17   13: . . .  UTCTime '000125161938Z'
 517 17   13: . . .  UTCTime '200120161938Z'
           : . . .  }
 532 30  112: . . . SEQUENCE {
 534 31   11: . . .  SET {
 536 30    9: . . . . SEQUENCE {
 538 06    3: . . . .  OBJECT IDENTIFIER countryName (2 5 4 6)
           : . . . . . (X.520 id-at (2 5 4))
 543 13    2: . . . .  PrintableString 'FR'
           : . . . .  }
           : . . . . }
 547 31   21: . . .  SET {
 549 30   19: . . . . SEQUENCE {
 551 06    3: . . . .  OBJECT IDENTIFIER organizationName (2 5 4 10)
           : . . . . . (X.520 id-at (2 5 4))
 556 13   12: . . . .  PrintableString 'EdelWeb S.A.'
           : . . . .  }
           : . . . . }
 570 31   40: . . .  SET {
 572 30   38: . . . . SEQUENCE {
 574 06    3: . . . .  OBJECT IDENTIFIER
           : . . . . . organizationalUnitName (2 5 4 11)
           : . . . . . (X.520 id-at (2 5 4))
```

```
 579 13 31: . . . .  PrintableString 'Clepsydre Demonstration Service'
            : . . . .  }
            : . . . . }
 612 31   32: . . .  SET {
 614 30   30: . . . . SEQUENCE {
 616 06    3: . . . .  OBJECT IDENTIFIER commonName (2 5 4 3)
            : . . . . . (X.520 id-at (2 5 4))
 621 13   23: . . . .  PrintableString 'Time Stamping Authority'
            : . . . .  }
            : . . . . }
            : . . .  }
 646 30  290: . . . SEQUENCE {
 650 30   13: . . .  SEQUENCE {
 652 06    9: . . . . OBJECT IDENTIFIER
            : . . . .  rsaEncryption (1 2 840 113549 1 1 1)
            : . . . .  (PKCS #1)
 663 05    0: . . . . NULL
            : . . . . }
 665 03  271: . . .  BIT STRING 0 unused bits
            : . . . . 30 82 01 0A 02 82 01 01 00 FA C3 17 AE EB B7 9D
            : . . . . EB AB BD 05 7E 39 43 6D 04 45 58 74 05 A5 CC F3
            : . . . . 6C 2F 8C 8E 77 7E C2 9F 12 11 5C 7D DB BE 23 28
            : . . . . 9A 90 D2 AB C6 A2 BA BD A3 7E 99 A6 99 21 A5 D8
            : . . . . 90 B9 CF A7 23 4E A0 56 A0 C1 0A 46 89 8E 3C 91
            : . . . . 67 37 FD 9B AB 49 17 FC 4A A5 F2 E4 4C 6E E3 6A
            : . . . . 1C 92 97 04 6F 7F 0C 5C FB 74 CB 95 7E 4C C3 58
            : . . . . 12 E8 A9 D6 F0 DD 12 44 15 E7 8B 2E AF 51 C0 0C
            : . . . . . . [ Another 142 bytes skipped ]
            : . . .  }
 940 A3  122: . . . [3] {
 942 30  120: . . .  SEQUENCE {
 944 30   15: . . . . SEQUENCE {
 946 06    3: . . . .  OBJECT IDENTIFIER basicConstraints (2 5 29 19)
            : . . . . . (X.509 id-ce (2 5 29))
 951 04    8: . . . .  OCTET STRING, encapsulates {
 953 30    6: . . . . .  SEQUENCE {
 955 01    1: . . . . . . BOOLEAN TRUE
 958 02    1: . . . . . . INTEGER 0
            : . . . . . . }
            : . . . . .  }
            : . . . .  }
 961 30   22: . . . . SEQUENCE {
 963 06    3: . . . .  OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
            : . . . . . (X.509 id-ce (2 5 29))
 968 01    1: . . . .  BOOLEAN TRUE
 971 04   12: . . . .  OCTET STRING, encapsulates {
 973 30   10: . . . . .  SEQUENCE {
 975 06    8: . . . . . . OBJECT IDENTIFIER '1 3 6 1 5 5 7 3 10'
```

```
            : . . . . . . }
            : . . . . .  }
            : . . . .  }
 985 30   77: . . . . SEQUENCE {
 987 06    8: . . . .  OBJECT IDENTIFIER
            : . . . . . authorityInfoAccess (1 3 6 1 5 5 7 1 1)
            : . . . . . (PKIX private extension)
 997 01    1: . . . .  BOOLEAN TRUE
1000 04   62: . . . .  OCTET STRING, encapsulates {
1002 30   60: . . . . .  SEQUENCE {
1004 30   58: . . . . . . SEQUENCE {
1006 06    8: . . . . . .  OBJECT IDENTIFIER '1 3 6 1 5 5 7 48 4'
1016 86   46: . . . . . .  [6]
            : . . . . .  'https://clepsydre.edelweb.fr/dvcs/service-ccpd'
            : . . . . . .  }
            : . . . . . . }
            : . . . . .  }
            : . . . .  }
            : . . . . }
            : . . .  }
            : . . . }
1064 30   13: . .  SEQUENCE {
1066 06    9: . . . OBJECT IDENTIFIER
            : . . .  md5withRSAEncryption (1 2 840 113549 1 1 4)
            : . . .  (PKCS #1)
1077 05    0: . . . NULL
            : . . . }
1079 03  257: . .  BIT STRING 0 unused bits
            : . . . 08 DA AF 5B 09 39 66 D3 BE 80 1D D7 72 B5 2C A3
            : . . . 04 FB 46 F8 05 F5 BF 83 F3 6D 6D 32 28 1C 46 EE
            : . . . 0F EA 30 61 8A 1E 8A 03 4E 98 81 60 1F 97 17 53
            : . . . D1 54 73 3F 72 98 45 D3 10 9A D3 77 B8 74 0E 9A
            : . . . 90 29 8E AC A4 EB D2 24 6D F6 21 1D 3F 52 8B 2C
            : . . . E6 92 E7 52 C6 54 93 91 BC 57 74 21 38 39 75 CD
            : . . . 30 49 54 13 94 6C FE F1 64 38 1F 5F 7D BB E0 3E
            : . . . A8 F1 28 1C F1 D9 28 FA 32 1E 3B 48 BF 5C 70 21
            : . . . . . [ Another 128 bytes skipped ]
            : . .  }
            : . . }
1340 31  699: .  SET {
1344 30  695: . . SEQUENCE {
1348 02    1: . .  INTEGER 1
1351 30  124: . .  SEQUENCE {
1353 30  112: . . . SEQUENCE {
1355 31   11: . . .  SET {
1357 30    9: . . . . SEQUENCE {
1359 06    3: . . . .  OBJECT IDENTIFIER countryName (2 5 4 6)
            : . . . . . (X.520 id-at (2 5 4))
```

```
1364 13    2: . . . .  PrintableString 'FR'
            : . . . .  }
            : . . . . }
1368 31   21: . . .  SET {
1370 30   19: . . . . SEQUENCE {
1372 06    3: . . . .  OBJECT IDENTIFIER organizationName (2 5 4 10)
            : . . . . . (X.520 id-at (2 5 4))
1377 13   12: . . . .  PrintableString 'EdelWeb S.A.'
            : . . . .  }
            : . . . . }
1391 31   40: . . .  SET {
1393 30   38: . . . . SEQUENCE {
1395 06    3: . . . .  OBJECT IDENTIFIER
            : . . . . . organizationalUnitName (2 5 4 11)
            : . . . . . (X.520 id-at (2 5 4))
1400 13 31: . . . .  PrintableString 'Clepsydre Demonstration Service'
            : . . . .  }
            : . . . . }
1433 31   32: . . .  SET {
1435 30   30: . . . . SEQUENCE {
1437 06    3: . . . .  OBJECT IDENTIFIER commonName (2 5 4 3)
            : . . . . . (X.520 id-at (2 5 4))
1442 13   23: . . . .  PrintableString 'Time Stamping Authority'
            : . . . .  }
            : . . . . }
            : . . .  }
1467 02    8: . . . INTEGER
            : . . .  00 94 88 25 72 35 27 50
            : . . . }
1477 30    9: . .  SEQUENCE {
1479 06    5: . . . OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
            : . . .  (OIW)
1486 05    0: . . . NULL
            : . . . }
1488 A0  276: . .  [0] {
1492 30   26: . . . SEQUENCE {
1494 06    9: . . .  OBJECT IDENTIFIER
            : . . . . contentType (1 2 840 113549 1 9 3)
            : . . . . (PKCS #9 (1 2 840 113549 1 9))
1505 31   13: . . .  SET {
1507 06   11: . . . . OBJECT IDENTIFIER
            : . . . .  id-ct-dvcsresponse (1 2 840 113549 1 9 16 1 8)
            : . . . .  (S/MIME Content Types (1 2 840 113549 1 9 16 1))
            : . . . . }
            : . . .  }
1520 30   28: . . . SEQUENCE {
1522 06    9: . . .  OBJECT IDENTIFIER
            : . . . . signingTime (1 2 840 113549 1 9 5)
```

```
          : . . . . (PKCS #9 (1 2 840 113549 1 9))
1533 31   15: . . .  SET {
1535 17   13: . . . . UTCTime '000417171619Z'
          : . . . . }
          : . . .  }
1550 30   35: . . . SEQUENCE {
1552 06    9: . . .  OBJECT IDENTIFIER
          : . . . . messageDigest (1 2 840 113549 1 9 4)
          : . . . . (PKCS #9 (1 2 840 113549 1 9))
1563 31   22: . . .  SET {
1565 04   20: . . . . OCTET STRING
          : . . . .  68 50 DC 90 20 2E C2 F0 55 15 7F 77 A9 A6 0C 34
          : . . . .  CC 13 06 FA
          : . . . . }
          : . . .  }
1587 30  178: . . . SEQUENCE {
1590 06   11: . . .  OBJECT IDENTIFIER
         : . . . id-aa-signingCertificate (1 2 840 113549 1 9 16 2 12)
      : . . (S/MIME Authenticated Attributes (1 2 840 113549 1 9 16 2))
1603 31  162: . . .  SET {
1606 30  159: . . . . SEQUENCE {
1609 30  156: . . . .  SEQUENCE {
1612 30  153: . . . . . SEQUENCE {
1615 04   20: . . . . .  OCTET STRING
          : . . . .  5C F1 18 F3 4A CA B4 67 D6 D8 E7 F8 3B 4A D9 7A
          : . . . .  32 A5 43 A5
1637 30  128: . . . . .  SEQUENCE {
1640 30  116: . . . . . . SEQUENCE {
1642 A4  114: . . . . . .  [4] {
1644 30  112: . . . . . . . SEQUENCE {
1646 31   11: . . . . . .  SET {
1648 30    9: . . . . . . . SEQUENCE {
1650 06    3: . . . . . .  OBJECT IDENTIFIER
          : . . . . . . . . countryName (2 5 4 6)
          : . . . . . . . . (X.520 id-at (2 5 4))
1655 13    2: . . . . . .  PrintableString 'FR'
          : . . . . . .  }
          : . . . . . . }
1659 31   21: . . . . . .  SET {
1661 30   19: . . . . . . SEQUENCE {
1663 06    3: . . . . . .  OBJECT IDENTIFIER
          : . . . . . . . . organizationName (2 5 4 10)
          : . . . . . . . . (X.520 id-at (2 5 4))
1668 13   12: . . . . . .  PrintableString 'EdelWeb S.A.'
          : . . . . . .  }
          : . . . . . . }
1682 31   40: . . . . . .  SET {
1684 30   38: . . . . . . SEQUENCE {
```

```
1686 06    3: . . . . . .  OBJECT IDENTIFIER
            : . . . . . . . organizationalUnitName (2 5 4 11)
            : . . . . . . . (X.520 id-at (2 5 4))
1691 13 31: . . . .PrintableString 'Clepsydre Demonstration Service'
            : . . . . . .  }
            : . . . . . . }
1724 31   32: . . . . . .  SET {
1726 30   30: . . . . . . SEQUENCE {
1728 06    3: . . . . . .  OBJECT IDENTIFIER
            : . . . . . . . commonName (2 5 4 3)
            : . . . . . . . (X.520 id-at (2 5 4))
1733 13 23: . . . . . .  PrintableString 'Time Stamping Authority'
            : . . . . . .  }
            : . . . . . . }
            : . . . . . .  }
            : . . . . . . }
            : . . . . . .  }
1758 02    8: . . . . . INTEGER
            : . . . .  00 94 88 25 72 35 27 50
            : . . . . . . }
            : . . . . .  }
            : . . . . . }
            : . . . .  }
            : . . . . }
            : . . .  }
            : . . . }
1768 30   13: . .  SEQUENCE {
1770 06    9: . . . OBJECT IDENTIFIER
            : . . .  rsaEncryption (1 2 840 113549 1 1 1)
            : . . .  (PKCS #1)
1781 05    0: . . . NULL
            : . . . }
1783 04  256: . .  OCTET STRING
            : . . . 2E 70 9F 56 5E 01 56 A9 E1 47 81 12 35 21 29 09
            : . . . 16 7A ED 45 F9 5A A2 ED E4 FE 9D 2C E4 DA 12 66
            : . . . 62 14 59 61 8B 50 7B 01 82 3D BD 7E E6 38 D0 A8
            : . . . A0 37 98 79 13 26 39 29 C6 72 20 A9 95 71 E7 53
            : . . . 7F 79 77 98 EF 23 02 4E B9 BD 90 9B AC 05 A2 70
            : . . . 8F 3A 42 36 9C 2C B0 94 B1 2B 0B 36 94 0E 78 0E
            : . . . B0 D1 09 20 63 BC FF CD 32 F1 5A D3 AB 9F 93 9C
            : . . . 5A A3 58 99 A0 28 11 E0 80 4D 4D 1E 77 04 F4 50
            : . . . . . [ Another 128 bytes skipped ]
            : . .  }
            : . . }
            : .  }
            : . }
            :  }
```

The corresponding data in PEM format (together with a technical textual description) are:

```
Data Validation Certificate:
    Request Information:
      Service: Certify Claim of Possession of Data - ccpd(4)
      Policy: EdelWeb Customer Policy Clepsydre
      Requester:
        DirName:/C=FR/L=Paris/O=EdelWeb/CN=Peter Sylvester
      DVCS:
        DirName:/C=FR/O=EdelWeb S.A./
  OU=Clepsydre Demonstration Service/CN=Time Stamping Authority
    SerialNumber: 01780a1eca8823
    MessageDigest:
      Algorithm: sha1
      Data     : 75B685AF6F89467DE80715251E45978FCD1FA566
    Asserted Time:
      Generalized Time: 17-Apr-2000 19:16:17 (Apr 17 17:16:17 2000 GMT)
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            94:88:17:17:64:37:32
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=FR, O=EdelWeb S.A.,
    OU=Clepsydre Demonstration Service, CN=Time Stamping Authority
        Validity
            Not Before: Jan 25 16:19:38 2000 GMT
            Not After : Jan 20 16:19:38 2020 GMT
        Subject: C=FR, O=EdelWeb S.A.,
    OU=Clepsydre Demonstration Service, CN=Time Stamping Authority
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:fa:c3:17:ae:eb:b7:9d:eb:ab:bd:05:7e:39:43:
                    6d:04:45:58:74:05:a5:cc:f3:6c:2f:8c:8e:77:7e:
                    c2:9f:12:11:5c:7d:db:be:23:28:9a:90:d2:ab:c6:
                    a2:ba:bd:a3:7e:99:a6:99:21:a5:d8:90:b9:cf:a7:
                    23:4e:a0:56:a0:c1:0a:46:89:8e:3c:91:67:37:fd:
                    9b:ab:49:17:fc:4a:a5:f2:e4:4c:6e:e3:6a:1c:92:
                    97:04:6f:7f:0c:5c:fb:74:cb:95:7e:4c:c3:58:12:
                    e8:a9:d6:f0:dd:12:44:15:e7:8b:2e:af:51:c0:0c:
                    5f:a8:65:fc:47:a1:c9:98:1f:d4:e1:ea:bc:1c:1a:
                    27:bb:8b:56:f1:12:55:10:f4:8e:d8:9f:19:9c:1e:
                    81:f7:db:63:dd:88:37:3f:71:79:5b:96:e2:5f:82:
                    d5:12:19:05:0d:e1:3d:a5:6d:66:e4:2c:1e:ed:c7:
                    4c:b8:df:aa:38:c8:15:6a:ae:25:7d:46:2a:07:f9:
```

```
                    83:77:c4:51:ee:90:dc:05:d0:c3:f0:f1:5f:e8:d4:
                    ed:5d:34:70:91:9d:9f:08:55:7d:5b:e5:8d:5f:35:
                    59:83:4e:72:19:bb:9c:88:d1:7a:fc:23:a5:84:99:
                    b4:17:8a:4d:6c:9d:d0:a6:35:80:5f:ca:fb:24:8b:
                    54:1d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE, pathlen:0
            X509v3 Extended Key Usage: critical
                DVCS Signing
            Authority Information Access: critical
         DVCS - URI:https://clepsydre.edelweb.fr/dvcs/service-ccpd

    Signature Algorithm: md5WithRSAEncryption
        08:da:af:5b:09:39:66:d3:be:80:1d:d7:72:b5:2c:a3:04:fb:
        46:f8:05:f5:bf:83:f3:6d:6d:32:28:1c:46:ee:0f:ea:30:61:
        8a:1e:8a:03:4e:98:81:60:1f:97:17:53:d1:54:73:3f:72:98:
        45:d3:10:9a:d3:77:b8:74:0e:9a:90:29:8e:ac:a4:eb:d2:24:
        6d:f6:21:1d:3f:52:8b:2c:e6:92:e7:52:c6:54:93:91:bc:57:
        74:21:38:39:75:cd:30:49:54:13:94:6c:fe:f1:64:38:1f:5f:
        7d:bb:e0:3e:a8:f1:28:1c:f1:d9:28:fa:32:1e:3b:48:bf:5c:
        70:21:29:ef:be:72:24:da:0d:f9:51:7a:fe:d7:f5:ff:e8:c2:
        ea:c6:4c:45:14:51:53:fd:00:d5:5b:cc:67:2a:23:94:31:9e:
        c2:90:38:9b:b0:df:f9:de:67:0c:57:5c:d7:b0:fc:f2:72:96:
        c4:d1:7a:9d:a0:e6:51:24:99:9e:89:c6:39:f9:72:7a:44:fd:
        2d:3f:bc:df:c7:25:27:94:a1:b5:7d:ba:06:75:67:1c:95:6c:
        bd:2c:74:41:3e:cd:cd:39:5c:2e:9c:c3:c3:09:e3:79:d5:eb:
        85:e8:f1:72:29:80:f6:c6:6e:61:1b:58:fc:87:3e:d9:e1:53:
        10:e0:b1:05
```

```
-----BEGIN PKCS7-----
MIIH9wYJKoZIhvcNAQcCoIIH6DCCB+QCAQMxCzAJBgUrDgMCGgUAMIIBLQYLKoZI
hvcNAQkQAQigggEcBIIBGDCCARQwgdYKAQSgTaRLMEkxCzAJBgNVBAYTAkZSMQ4w
DAYDVQQHEwVQYXJpczEQMA4GA1UEChMHRWRlbFdlYjEYMBYGA1UEAxMPUGV0ZXIg
U3lsdmVzdGVyoQwGCisGAQQBqT0BAgGidKRyMHAxCzAJBgNVBAYTAkZSMRUwEwYD
VQQKEwxFZGVsV2ViIFMuQS4xKDAmBgNVBAsTH0NsZXBzeWRyZSBEZW1vbnN0cmF0
aW9uIFNlcnZpY2UxIDAeBgNVBAMTF1RpbWUgU3RhbXBpbmcgQXV0aG9yaXR5MB8w
BwYFKw4DAhoEFHW2ha9viUZ96AcVJR5Fl4/NH6VmAgcBeAoeyogjGA8yMDAwMDQx
NzE3MTYxN1qgggPgMIID3DCCAsSgAwIBAgIIAJSIFxdkNzIwDQYJKoZIhvcNAQEE
BQAwcDELMAkGA1UEBhMCRlIxFTATBgNVBAoTDEVkZWxXZWIgUy5BLjEoMCYGA1UE
CxMfQ2xlcHN5ZHJlIERlbW9uc3RyYXRpb24gU2VydmljZTEgMB4GA1UEAxMXVGlt
ZSBTdGFtcGluZyBBdXRob3JpdHkwHhcNMDAwMTI1MTYxOTM4WhcNMjAwMTIwMTYx
OTM4WjBwMQswCQYDVQQGEwJGUjEVMBMGA1UEChMMRWRlbFdlYiBTLkEuMSgwJgYD
VQQLEx9DbGVwc3lkcmUgRGVtb25zdHJhdGlvbiBTZXJ2aWNlMSAwHgYDVQQDExdU
aW1lIFN0YW1waW5nIEF1dGhvcml0eTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAPrDF67rt53rq70FfjlDbQRFWHQFpczzbC+Mjnd+wp8SEVx9274jKJqQ
0qvGorq9o36ZppkhpdiQuc+nI06gVqDBCkaJjjyRZzf9m6tJF/xKpfLkTG7jahyS
```

lwRvfwxc+3TLlX5Mw1gS6KnW8N0SRBXniy6vUcAMX6hl/EehyZgf1OHqvBwaJ7uL
VvESVRD0jtifGZwegffbY92INz9xeVuW4l+C1RIZBQ3hPaVtZuQsHu3HTLjfqjjI
FWquJX1GKgf5g3fEUe6Q3AXQw/DxX+jU7V00cJGdnwhVfVvljV81WYNOchm7nIjR
evwjpYSZtBeKTWyd0KY1gF/K+ySLVB0CAwEAAaN6MHgwDwYDVR0TBAgwBgEB/wIB
ADAWBgNVHSUBAf8EDDAKBggrBgEFBQcDCjBNBggrBgEFBQcBAQEB/wQ+MDwwOgYI
KwYBBQUHMASGLmh0dHBzOi8vY2xlcHN5ZHJlLmVkZWx3ZWIuZnIvZHZjcy9zZXJ2
aWNlLWNjcGQwDQYJKoZIhvcNAQEEBQADggEBAAjar1sJOWbTvoAd13K1LKME+0b4
BfW/g/NtbTIoHEbuD+owYYoeigNOmIFgH5cXU9FUcz9ymEXTEJrTd7h0DpqQKY6s
pOvSJG32IR0/Uoss5pLnUsZUk5G8V3QhODl1zTBJVBOUbP7xZDgfX3274D6o8Sgc
8dko+jIeO0i/XHAhKe++ciTaDflRev7X9f/owurGTEUUUVP9ANVbzGcqI5QxnsKQ
OJuw3/neZwxXXNew/PJylsTRep2g5lEkmZ6Jxjn5cnpE/S0/vN/HJSeUobV9ugZ1
ZxyVbL0sdEE+zc05XC6cw8MJ43nV64Xo8XIpgPbGbmEbWPyHPtnhUxDgsQUxggK7
MIICtwIBATB8MHAxCzAJBgNVBAYTAkZSMRUwEwYDVQQKEwxFZGVsV2ViIFMuQS4x
KDAmBgNVBAsTH0NsZXBzeWRyZSBEZW1vbnN0cmF0aW9uIFNlcnZpY2UxIDAeBgNV
BAMTF1RpbWUgU3RhbXBpbmcgQXV0aG9yaXR5AggAlIglcjUnUDAJBgUrDgMCGgUA
oIIBFDAaBgkqhkiG9w0BCQMxDQYLKoZIhvcNAQkQAQgwHAYJKoZIhvcNAQkFMQ8X
DTAwMDQxNzE3MTYxOVowIwYJKoZIhvcNAQkEMRYEFGhQ3JAgLsLwVRV/d6mmDDTM
Ewb6MIGyBgsqhkiG9w0BCRACDDGBojCBnzCBnDCBmQQUXPEY80rKtGfW2Of4O0rZ
ejKlQ6UwgYAwdKRyMHAxCzAJBgNVBAYTAkZSMRUwEwYDVQQKEwxFZGVsV2ViIFMu
QS4xKDAmBgNVBAsTH0NsZXBzeWRyZSBEZW1vbnN0cmF0aW9uIFNlcnZpY2UxIDAe
BgNVBAMTF1RpbWUgU3RhbXBpbmcgQXV0aG9yaXR5AggAlIglcjUnUDANBgkqhkiG
9w0BAQEFAASCAQAucJ9WXgFWqeFHgRI1ISkJFnrtRflaou3k/p0s5NoSZmIUWWGL
UHsBgj29fuY40KigN5h5EyY5KcZyIKmVcedTf3l3mO8jAk65vZCbrAWicI86Qjac
LLCUsSsLNpQOeA6w0QkgY7z/zTLxWtOrn5OcWqNYmaAoEeCATU0edwT0UAfVi1Sg
IzL/ppziurjbVUfJyLoH75AUSKi2xXzVqSB0HFbvjxuz/IdtgfHUbxqHMJJHaeB5
4LwQmc9NNkw2A1Fy0VumHi2G8R8K6L/rOPnOGuywj1GuKjtGhL9NjJ/uH+/FNaNj
vjjAA3w6XrjPOxgQiNu7T3j2++QcjdT4++tQ
-----END PKCS7-----

Appendix G - Acknowledgements

This document is based on two initial works from Robert Zuccherato and Carlisle Adams, both at Entrust Technologies, for time stamping and for notary and data certification services.

Thanks to Denis Pinkas, Bull and Bruno Salgueiro, SIBS for valuable comments.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.